Name: Mohit Ranjit More

Class:TE-IT-T3

Subject: Operating System

Roll No: 3014

Pune Vidyarthi Griha's
College of Engineering and Technology & G. K. Pate
(Wani) Institute of Management
Approved by AICTE, DTE (Code: 6274) | Affiliated to SPPU, Pune | NAAC Second Cycle 'A' Grade

Assignment No: 7a

Problem Statement:

FIFOS: Full duplex communication between two independent processes. First process accepts sentences and writes on one pipe to be read by second process and second process counts number characters, number of words and number of lines in accepted sentences, writes this output in a text file and writes the contents of the file on second pipe to be read by first process and displays on standard
output.

Solution:

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>

#define PIPE1 "/tmp/pipe1"
#define PIPE2 "/tmp/pipe2"
#define BUFFER_SIZE 1024

// Function to count characters, words, and lines
void count_chars_words_lines(char *text, int *chars, int *words, int *lines) {
   *chars = *words = *lines = 0;
   int in_word = 0;

   for (int i = 0; text[i] != '\0'; i++) {
      (*chars)++;
      if (text[i] == '\n') (*lines)++;
      if (text[i] == ' ' || text[i] == '\n' || text[i] == '\t') {
         in_word = 0;
      } else if (in_word == 0) {
         in_word = 1;
         (*words)++;
      }
   }
}

// Process 1: Accept sentences, send to pipe1, receive result from pipe2
void process1() {
   char input[BUFFER_SIZE], result[BUFFER_SIZE];
```

```
    // Open pipe1 for writing
    int fd1 = open(PIPE1, O_WRONLY);
    if (fd1 < 0) {
        perror("Error opening pipe1 for writing");
        exit(1);
    }

    // Get sentence input from user
    printf("Enter a sentence: ");
    fgets(input, BUFFER_SIZE, stdin);

    // Write input to pipe1
    write(fd1, input, strlen(input) + 1);
    close(fd1);

    // Open pipe2 for reading
    int fd2 = open(PIPE2, O_RDONLY);
    if (fd2 < 0) {
        perror("Error opening pipe2 for reading");
        exit(1);
    }

    // Read the result from pipe2 and display it
    read(fd2, result, BUFFER_SIZE);
    printf("Result from Process 2:\n%s", result);
    close(fd2);
}

// Process 2: Read from pipe1, process the text, and send result back via pipe2
void process2() {
    char input[BUFFER_SIZE], result[BUFFER_SIZE];
    int chars, words, lines;

    // Open pipe1 for reading
    int fd1 = open(PIPE1, O_RDONLY);
    if (fd1 < 0) {
        perror("Error opening pipe1 for reading");
        exit(1);
    }

    // Read the sentence from pipe1
    read(fd1, input, BUFFER_SIZE);
    close(fd1);

    // Process the input: count characters, words, and lines
    count_chars_words_lines(input, &chars, &words, &lines);

    // Prepare the result string
    snprintf(result, BUFFER_SIZE, "Characters: %d\nWords: %d\nLines: %d\n", chars, words, lines);

    // Open pipe2 for writing
```

```
    int fd2 = open(PIPE2, O_WRONLY);
    if (fd2 < 0) {
        perror("Error opening pipe2 for writing");
        exit(1);
    }

    // Write the result to pipe2
    write(fd2, result, strlen(result) + 1);
    close(fd2);
}

int main() {
    // Create named pipes (FIFOs)
    mkfifo(PIPE1, 0666);
    mkfifo(PIPE2, 0666);

    pid_t pid = fork();  // Fork the process

    if (pid > 0) {
        // Parent process (Process 1)
        process1();
    } else if (pid == 0) {
        // Child process (Process 2)
        process2();
    } else {
        perror("Fork failed");
        exit(1);
    }

    // Remove the named pipes after communication
    unlink(PIPE1);
    unlink(PIPE2);

    return 0;
}
```

Source Output

```
mohit@mohit-VirtualBox:~$ cd mohit
mohit@mohit-VirtualBox:~/mohit$ gedit fifo_comm.c
mohit@mohit-VirtualBox:~/mohit$ gcc fifo_comm.c -o fifo_comm
mohit@mohit-VirtualBox:~/mohit$ ./fifo_comm
Enter a sentence: Hello World!
Result from Process 2:
Characters: 13
Words: 2
Lines: 1
mohit@mohit-VirtualBox:~/mohit$
```